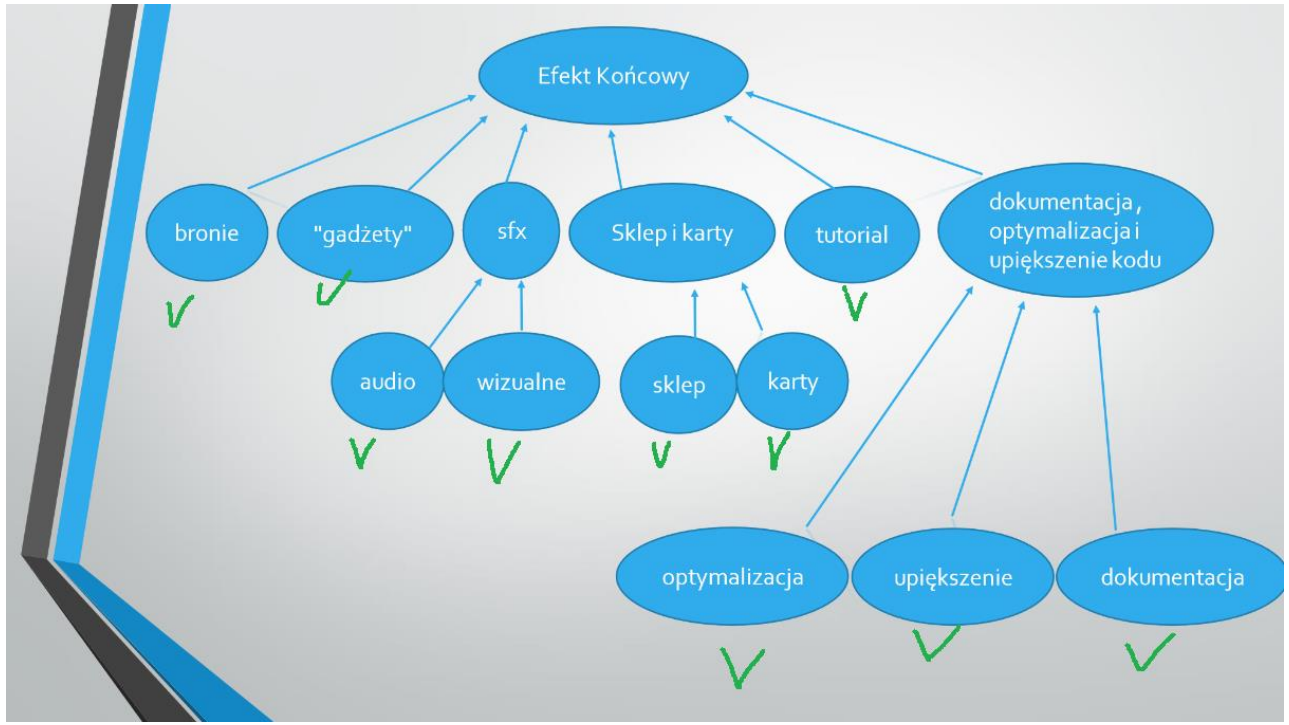
Projekt w języku skryptowym

KACPER ABRAM

Opis Projektu

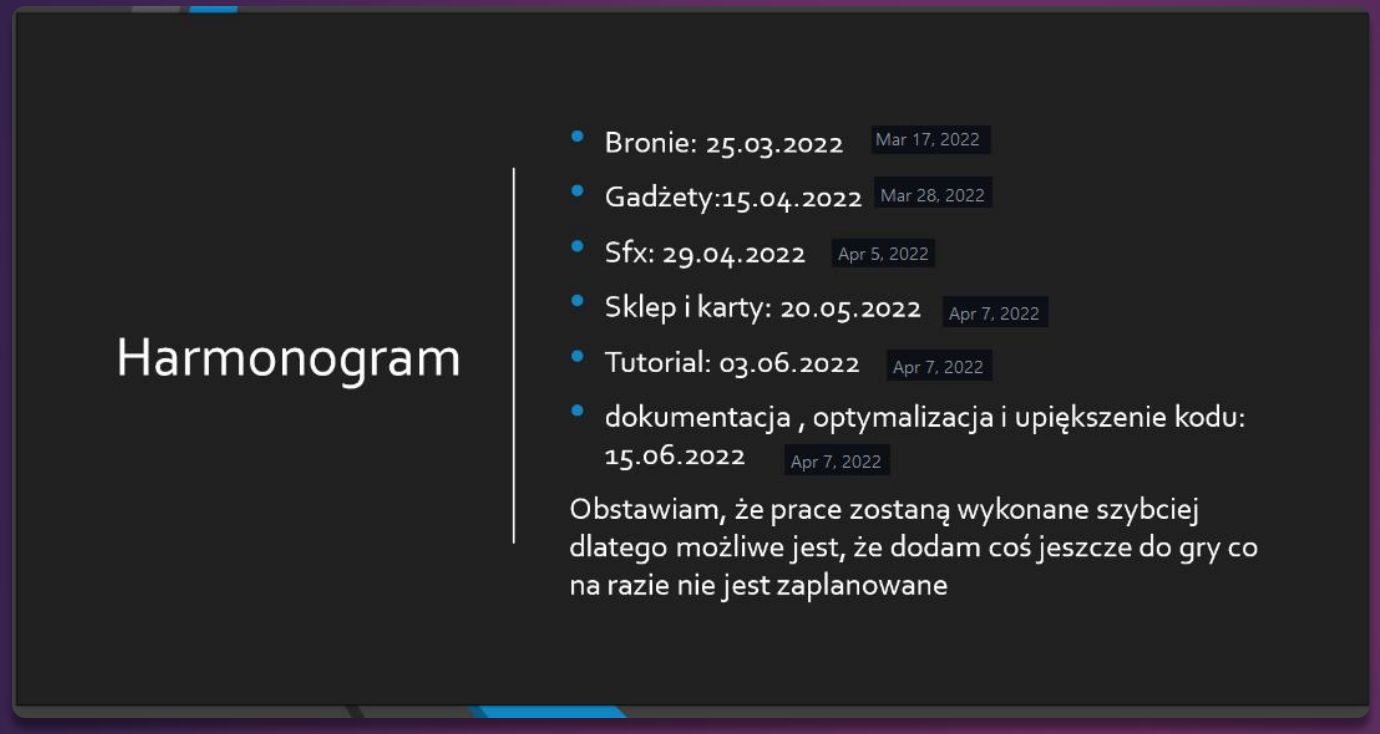
- Projektem była gra turowa, w której celem jest przejść jak najwięcej poziomów. Odbywa się ona na kwadratowej planszy 15x15. Grafika składa się głównie z spritów 32px x 32px w stylu pixelartowym. Gra zawiera efekty dźwiękowe.

Co udało się osiągnąć



Wszystko co było założone

Harmonogram



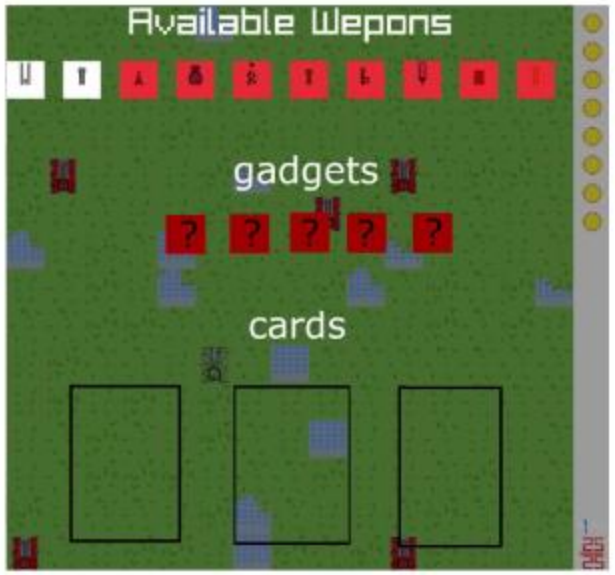
*Dokumentacja itp dopiero pulla zrobiłem 20.05 bo zapomniałem o nim i usunąłem mały błąd przy okazji

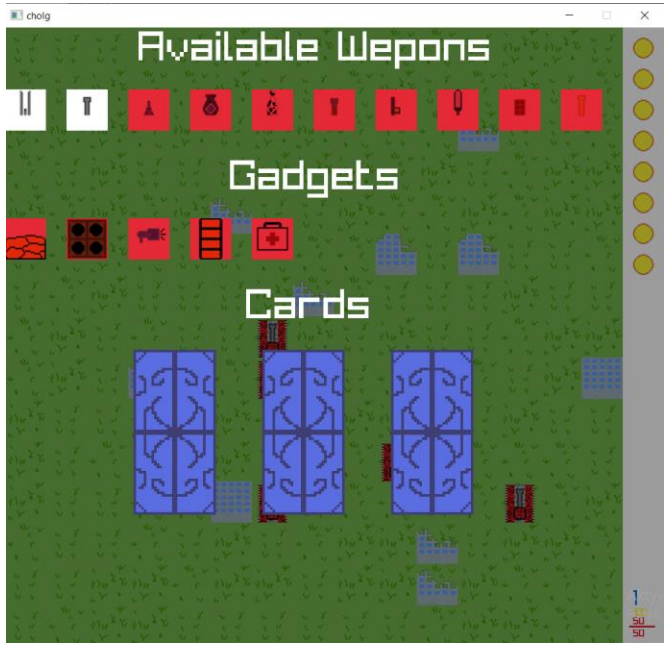
Zmiany względem planowanego rezultatu

- Brak

Planowany rezultat w porównaniu do gotowego projektu

Menu(eq)





Prawie wszystko jest obiektami

```python
class trawa:
    def __init__(self,posx,posy):
        self.x=posx
        self.y=posy
        losowa = random.randint(0,2)
        match losowa:
            case 0:
                self.sprite = trawa_tile1
            case 1:
                self.sprite = trawa_tile2
            case 2:
                self.sprite = trawa_tile3
    def narysuj(self):
        pr.draw_texture(self.sprite,self.x*64,self.y*64,pr.WHITE)
```

```python
class gracz:
    def __init__(self,posx,posy): ···
    def narysuj(self): ···
    def strzel(self): ···
    def zadaj_obrazenia(self,obrazenia): ···
    def uzycie_gadzetu(self): ···
```

```python
class przeciwnik:
    def __init__(self,posx,posy,tury_PODANE):
        self.x = posx
        self.y = posy
        self.obrot = 0
        self.hp = 10
        self.aktualna_bron = "normal_barrel"
        self.tury = tury_PODANE

    def narysuj(self): ···
    def zadaj_obrazenia(self,obrazenia): ···
    def tura_przeciwnika(self): #https://www.yout
```

```python
class przeszkody:
    def __init__(self,posx,posy,rodzaj):
        self.x=posx
        self.y=posy
        match rodzaj:
            case 0:
                self.hp = 10
                self.sprite = budynek_tile1
            case 1:
                self.hp = 7
                self.sprite = budynek_tile2
            case 2:
                self.hp = 3
                self.sprite = budynek_tile3
            case "sand_bag":
                self.hp = 30
                self.sprite = ui_sandbags_texture
    def narysuj(self): ···
    def zadaj_obrazenia(self,obrazenia): ···
```

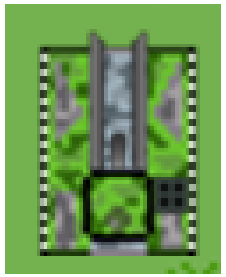
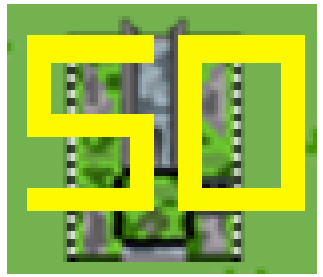
```python
class dash_smoke:
    def __init__(self,posx,posy,obrotval): ···
    def narysuj(self): ···

class Explosion:
    def __init__(self,posx,posy): ···
    def narysuj(self): ···

class Damage:
    def __init__(self,posx,posy): ···
    def narysuj(self): ···

class Barrel_smoke:
    def __init__(self,posx,posy,rotation): ···
    def narysuj(self): ···

class Laser:
    def __init__(self,posx,posy,rotation): ···
    def narysuj(self): ···

class Heling_animation:
    def __init__(self,posx,posy): ···
    def narysuj(self): ···

class Battery_animation:
    def __init__(self,posx,posy): ···
    def narysuj(self): ···
```

*żeczy takie jak zmienna które sprawdzają na jakim poziomie jest gracz są zmiennymi globalnymi

Sposób renderowania gracza

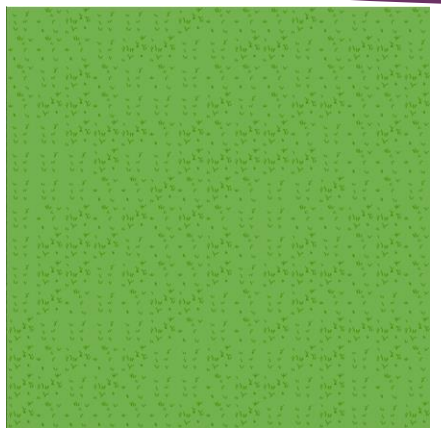
```
def narysuj(self):
    pr.draw_texture_tiled(czolg,pr.Rectangle(0,0,64,64),pr.Rectangle(self.x*64+32-(self.animacja*4)*(self.obrot%4==1)+(self.animacja*4)*(self.obrot%4==3) ,self.y*64+
    Rysowanie_Broni(self.aktualna_bron,self.x-((self.obrot%4==1)*self.animacja/16)+((self.obrot%4==3)*self.animacja/16),self.y+((self.obrot%4==0)*self.animacja/16)-(
    Rysowanie_Gadzetu(self.aktualny_gadzet,self.x-((self.obrot%4==1)*self.animacja/16)+((self.obrot%4==3)*self.animacja/16),self.y+((self.obrot%4==0)*self.animacja/1
    if (self.animacja >0):
        self.animacja -=1
    if(self.animacja <0):
        self.animacja +=1
    if (pr.get_mouse_x() > self.x*64 and pr.get_mouse_x() < (self.x+1)*64 and pr.get_mouse_y() > self.y*64 and pr.get_mouse_y() < (self.y+1)*64 and menu_eq == False)
            pr.draw_text(str(self.hp),self.x*64,self.y*64,64,pr.YELLOW)
```

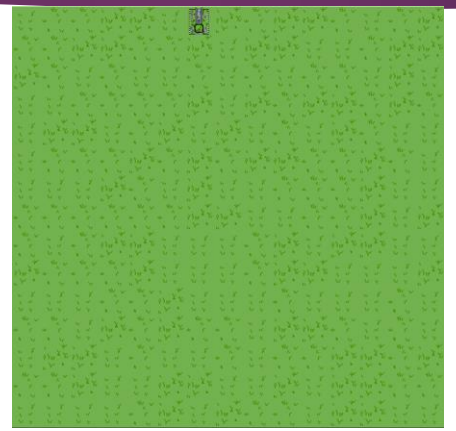


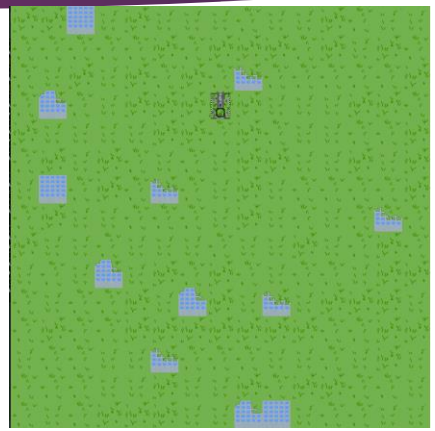



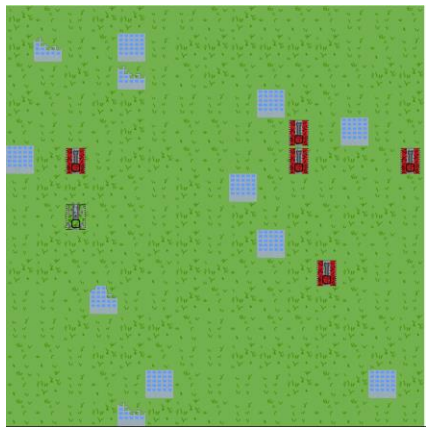
Sposób renderowania obrazu

```
def RysowaniePlanszy():
    for obj in trawy_arr:
        obj.narysuj()
    obiekt_gracz.narysuj()
    for obj in przeszkody_arr:
        obj.narysuj()
    for obj in przeciwnicy_arr:
        obj.narysuj()
    for obj in smoke_arr:
        obj.narysuj()
    for obj in particles_arr:
        obj.narysuj()
```

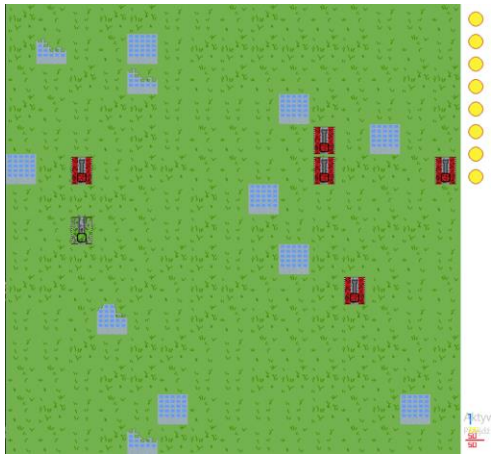








```
def RysowanieUi():
    global obiekt_gracz
    for i in range(obiekt_gracz.maxenergii):
        if(i<obiekt_gracz.aktualnaenergia):
            pr.draw_texture(energy_full_texture,976,16+i*48,pr.WHITE)
        else:
            pr.draw_texture(energy_empty_texture,976,16+i*48,pr.WHITE)

    pr.draw_text(str(obiekt_gracz.maxhp),976,932,16,pr.RED)
    pr.draw_rectangle(970,928,40,3,pr.RED)
    pr.draw_text(str(obiekt_gracz.hp),976,913,16,pr.RED)
    pr.draw_text(str(poziom),976,870,32,pr.BLUE)
    pr.draw_text(str(obiekt_gracz.money),976,900,16,pr.YELLOW)
    if menu_eq == True:
```



Przeciwnicy

```python
def tura_przeciwnika(self): #https://www.youtube.com/watch?v=8SigT_jhz4I - tutorial z jakiego kozystalem do pathfindingu
    czy_strzelono = False #czolg nie poruszy sie w tej tuze gdy strzeli, nie wazne czy to 1 czy 2 czy 3 ruch
    for i in range(self.tury):
        matrix =[[1 for x in range(15)] for y in range(15)]
        for obj in przeciwnicy_arr:

            matrix[obj.y][obj.x] = 0
        for obj in przeszkody_arr:
            matrix[obj.y][obj.x] = 0
        grid = Grid(matrix = matrix)
        start = grid.node(self.x,self.y)
        end = grid.node(obiekt_gracz.x,obiekt_gracz.y)
        finder = AStarFinder()
        path,runs = finder.find_path(start,end,grid)
        if len(path)>1 and czy_strzelono == False:
            next_x = path[1][0]
            next_y = path[1][1]
            kierunek = ''
            if(self.x == next_x+1 and self.y == next_y):
                kierunek = "lewo"
            elif(self.x == next_x-1 and self.y == next_y):
                kierunek = "prawo"
            elif(self.x == next_x and self.y == next_y+1):
                kierunek = "gora"
            elif(self.x == next_x and self.y == next_y-1):
                kierunek = "dol"
```

```python
elif(self.x == next_x and self.y == next_y-1):
    kierunek = "dol"

match self.obrot:
    case 0:
        if kierunek == "gora":
            if [obiekt_gracz.x,obiekt_gracz.y]in[[self.x,self.y-1],[self.x,self.y-2],[self.x,self.y-3]]:
                czy_strzelono = True
                strzal(self.x,self.y,self.obrot,self.aktualna_bron)
            elif obiekt_gracz.y != self.y-1 or obiekt_gracz.x !=self.x:
                if CzyJestobiekt(next_x,next_y) == False:
                    self.y -= 1
        elif kierunek == "prawo" or kierunek == "dol":
            self.obrot+=1
            if(self.obrot>=4):
                self.obrot=0
        elif kierunek == "lewo":
            self.obrot-=1
            if(self.obrot<0):
                self.obrot=3
```

Przeciwnicy w skrócie

- Co turę ma 3 ruchy, jeżeli strzeli kończy turę, używa A* żeby znaleść drogę do gracza, jeżeli gracz jest w zasięgu strzału (nawet jeżeli jest za budynkiem/przeciwnikiem) to strzeli do niego, jeżeli nie to spróbuje się do niego dostać

Bronie


```
def strzal(poczatekx,poczateky,kierunek,nazwa):
    targets = []
    pr.play_sound(dict_dzwiekow[nazwa])
    match nazwa:
        case "railgun_barrel": #zasieg 4 na wprzod , 4 obr , wszystkie cele
```

```
case "railgun_barrel": #zasieg 4 na wprzod , 4 obr , wszystkie cele
    for i in range(4):
        targets.append([poczatekx+(i+1)*(kierunek%4==1)-(i+1)*(kierunek%4==3),poczateky-(i+1)*(kierunek%4==0)+(i+1)*(kierunek%4==2)])
        Laser(poczatekx+(i+1)*(kierunek%4==1)-(i+1)*(kierunek%4==3),poczateky-(i+1)*(kierunek%4==0)+(i+1)*(kierunek%4==2),kierunek)

    for i in targets:
        if obiekt_gracz.x == i[0] and obiekt_gracz.y == i[1]:
            obiekt_gracz.zadaj_obrazenia(4)
        for obj in przeciwnicy_arr:
            if obj.x == i[0] and obj.y ==i[1]:
                obj.zadaj_obrazenia(4)
        for obj in przeszkody_arr:
            if obj.x == i[0] and obj.y == i[1]:
                obj.zadaj_obrazenia(4)
```

```
koszty_strzalu = {"railgun_barrel":3,"normal_barrel":3,"prisma_barrel":4,
```

```
dict_dzwiekow = {"railgun_barrel":railgun_shot_audio,"normal_barrel":normal_shot_audio,
```

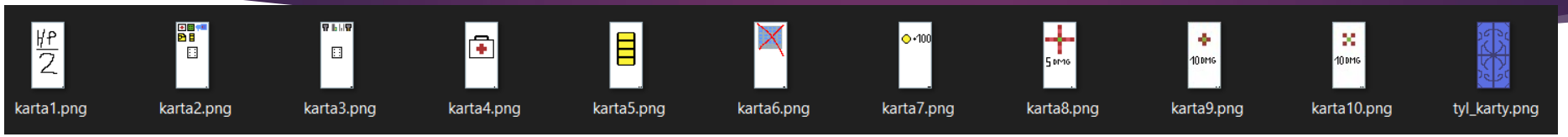
Gadżety


```
self.posiadane_gadzety = [] #"sand_bag","rocket_luncher","sound_wave","battery","medkit"
```

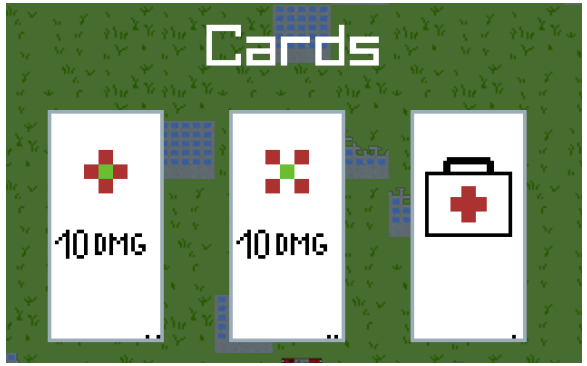
```
def Gadzet(nazwa):
    global obiekt_gracz
    global przeszkody_arr
    global przeciwnicy_arr
    aktualny_x = math.floor(pr.get_mouse_x()/64)
    aktualny_y = math.floor(pr.get_mouse_y()/64)
    match nazwa:
        case "sand_bag":
            if(CzyJestobiekt(aktualny_x,aktualny_y)):
                pr.play_sound(ui_cancel_audio)
                obiekt_gracz.aktualnaenergia += koszty_gadzetow["sand_bag"]
            else:
                przeszkody_arr.append(przeszkody(aktualny_x,aktualny_y,"sand_bag"))
                pr.play_sound(sandbag_medkit_battery_audio)
                obiekt_gracz.aktualny_gadzet = ""
                obiekt_gracz.posiadane_gadzety.remove("sand_bag")
```

```
def uzycie_gadzetu(self):
    if self.aktualny_gadzet in koszty_gadzetow:
        if self.aktualnaenergia >= koszty_gadzetow[self.aktualny_gadzet]:
            self.aktualnaenergia -=koszty_gadzetow[self.aktualny_gadzet]
            Gadzet(self.aktualny_gadzet)
```

Karty




```
self.posiadane_karty = []
```



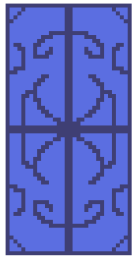

```
def uzycie_karty(nazwa):
    global obiekt_gracz
    global przeszkody_arr
    global przeciwnicy_arr
    obiekt_gracz.posiadane_karty.remove(nazwa)
    match nazwa:
        case "tyl": #na wszelki wypadek
            pr.play_sound(ui_cancel_audio)
        case "1":
            obiekt_gracz.zadaj_obrazenia(math.floor(obiekt_gracz.hp/2))
            for obj in przeszkody_arr:
                obj.zadaj_obrazenia(math.floor(obj.hp/2))
            for obj in przeciwnicy_arr:
                obj.zadaj_obrazenia(math.floor(obj.hp/2))
```

Sklep

cost: 100

732

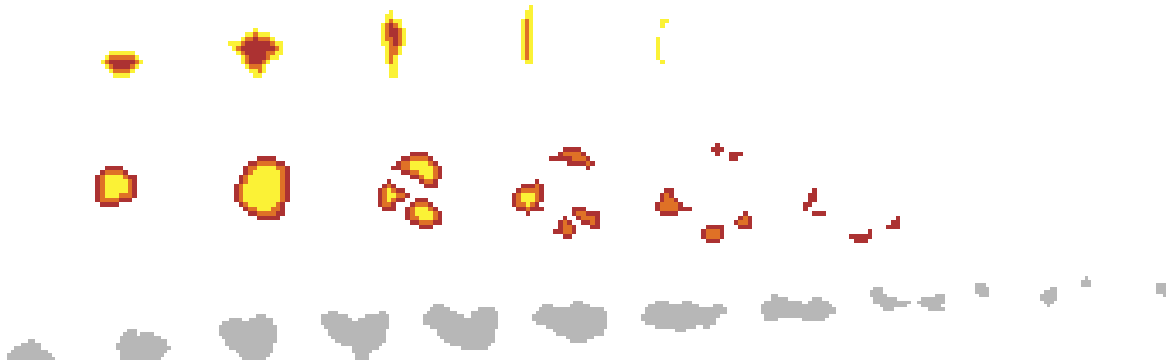
cost: 50

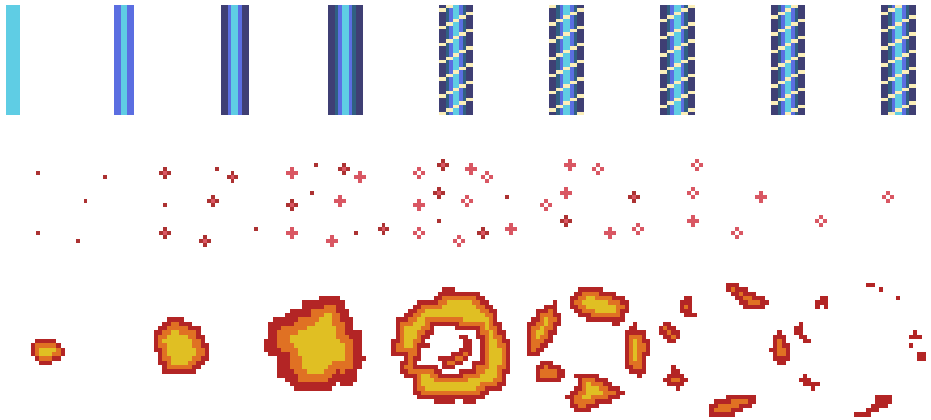


cost: 100

Ak
Prz

Particle,animacje itp





Koniec

- Jakieś pytania?